

Docket No. 219061US2SRD

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Noritaka KAWAKATSU

GAU:

SERIAL NO: NEW APPLICATION

EXAMINER:

FILED: HERewith

FOR: METHOD AND COMPUTER PROGRAM PRODUCT FOR SOFTWARE/HARDWARE LANGUAGE
MODEL CONVERSION

3-11-03
10/059185
10/059185
01/31/02

REQUEST FOR PRIORITY

ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

SIR:

- ☐ Full benefit of the filing date of U.S. Application Serial Number , filed , is claimed pursuant to the provisions of 35 U.S.C. §120.
- ☐ Full benefit of the filing date of U.S. Provisional Application Serial Number , filed , is claimed pursuant to the provisions of 35 U.S.C. §119(e).
- ☒ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of 35 U.S.C. §119, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY
Japan

APPLICATION NUMBER
2001-024887

MONTH/DAY/YEAR
January 31, 2001

Certified copies of the corresponding Convention Application(s)

- ☒ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee
- ☐ were filed in prior application Serial No. filed
- ☐ were submitted to the International Bureau in PCT Application Number
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.
- ☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and
- ☐ (B) Application Serial No.(s)
☐ are submitted herewith
☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.

Marvin J. Spivak

Registration No. 24,913

James D. Hamilton
Registration No. 28,421



22850

Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 10/98)

0151543

日 本 国 特 許 庁
JAPAN PATENT OFFICE

J0978 U.S. PRO
10/059185



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日
Date of Application:

2001年 1月31日

出 願 番 号
Application Number:

特願2001-024887

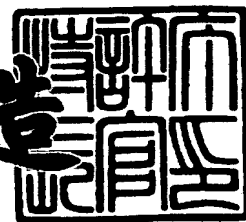
出 願 人
Applicant(s):

株式会社東芝

2001年12月14日

特 許 庁 長 官
Commissioner,
Japan Patent Office

及 川 耕 造



出証番号 出証特2001-3108900

【書類名】 特許願

【整理番号】 A000100062

【提出日】 平成13年 1月31日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 15/00

【発明の名称】 ソフトウェア・ハードウェア言語モデル変換装置及び方法並びにシステム設計支援装置

【請求項の数】 17

【発明者】

 【住所又は居所】 神奈川県川崎市幸区小向東芝町1番地 株式会社東芝研究開発センター内

 【氏名】 川勝 則孝

【特許出願人】

 【識別番号】 000003078

 【氏名又は名称】 株式会社 東芝

【代理人】

 【識別番号】 100058479

 【弁理士】

 【氏名又は名称】 鈴江 武彦

 【電話番号】 03-3502-3181

【選任した代理人】

 【識別番号】 100084618

 【弁理士】

 【氏名又は名称】 村松 貞男

【選任した代理人】

 【識別番号】 100068814

 【弁理士】

 【氏名又は名称】 坪井 淳

【選任した代理人】

【識別番号】 100092196

【弁理士】

【氏名又は名称】 橋本 良郎

【選任した代理人】

【識別番号】 100091351

【弁理士】

【氏名又は名称】 河野 哲

【選任した代理人】

【識別番号】 100088683

【弁理士】

【氏名又は名称】 中村 誠

【選任した代理人】

【識別番号】 100070437

【弁理士】

【氏名又は名称】 河井 将次

【手数料の表示】

【予納台帳番号】 011567

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 ソフトウェア・ハードウェア言語モデル変換装置及び方法
並びにシステム設計支援装置

【特許請求の範囲】

【請求項1】

ソフトウェア記述言語により記述された第1のモデルをハードウェア記述言語により記述された第2のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置であって、

同一の共有変数へ書き込みを行う複数の並列手続きが前記第1のモデルに含まれているか否かを考慮せずに、前記第1のモデルを前記第2のモデルに変換する変換手段と、

この変換手段により得られた前記第2のモデルに、前記同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検出手段と、

前記検出手段により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、前記共有変数の値を保持する信号へ出力する値解決プロセスを生成する値解決プロセス生成手段とを備えたことを特徴とするソフトウェア・ハードウェア言語モデル変換装置。

【請求項2】

前記値解決プロセス生成手段は、前記変換手段により得られた前記第2のモデルがコンポーネント階層を持つものである場合、前記値解決プロセスを該コンポーネント階層に応じて階層毎に生成することを特徴とする請求項1に記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項3】

前記値解決プロセス生成手段は、前記同一の共有変数に対する値解決プロセスが複数階層に存在する場合に、下位に位置する値解決プロセスについては、該値解決プロセスから、前記データ信号に加えて、該値解決プロセスから出力する該

データ信号の値が更新されるときに変化する代入タイミング信号をも出力させ、該データ信号及び該代入タイミング信号を上位の値解決プロセス接続することを特徴とする請求項 2 に記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項 4】

前記値解決プロセス生成手段は、前記同一の共有変数に対する値解決プロセスが複数階層に存在する場合に、最も上位に位置する値解決プロセスのみを残して他の値解決プロセスを削除し、削除した値解決プロセスに接続されていた前記並列プロセスからの前記データ信号及び該代入タイミング信号は、該最も上位に位置する値解決プロセスに接続することを特徴とする請求項 2 または 3 に記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項 5】

前記値解決プロセス生成手段により生成された前記値解決プロセスのうち、それが対象とするプロセスが、同一の手続きを呼び出す複数の並列手続きに相当する複数の並列プロセスであるものについては、該値解決プロセスを、各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する手続き呼び出し解決プロセスに変換するプロセス変換手段を更に備えたことを特徴とする請求項 1 ないし 4 のいずれか 1 項に記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項 6】

ソフトウェア記述言語により記述された第 1 のモデルをハードウェア記述言語により記述された第 2 のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置であって、

同一の手続きを呼び出す複数の並列手続きが前記第 1 のモデルに含まれているか否かを考慮せずに、前記第 1 のモデルを前記第 2 のモデルに変換する変換手段と、

この変換手段により得られた前記第 2 のモデルに、前記同一の手続きを呼び出

す複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検出手段と、

前記検出手段により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する手続き呼び出し解決プロセスを生成する手続き呼び出し解決プロセス生成手段とを備えたことを特徴とするソフトウェア・ハードウェア言語モデル変換装置。

【請求項 7】

前記手続き呼び出し解決プロセス生成手段は、前記変換手段により得られた前記第 2 のモデルがコンポーネント階層を持つものである場合、前記手続き呼び出し解決プロセスを該コンポーネント階層に応じて階層毎に生成することを特徴とする請求項 6 に記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項 8】

前記手続き呼び出し解決プロセス生成手段は、同一のプロセスに対する手続き呼び出し解決プロセスが複数階層に存在する場合に、下位に位置する手続き呼び出し解決プロセスを、上位の手続き呼び出し解決プロセスに接続することを特徴とする請求項 7 に記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項 9】

前記手続き呼び出し解決プロセス生成手段は、同一のプロセスに対する手続き呼び出し解決プロセスが複数階層に存在する場合に、最も上位に位置する手続き呼び出し解決プロセスのみを残して他の手続き呼び出し解決プロセスを削除し、削除した手続き呼び出し解決プロセスに接続されていた前記並列プロセスを、該最も上位に位置する手続き呼び出し解決プロセスに接続することを特徴とする請求項 7 または 8 に記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項 10】

ソフトウェア記述言語により記述された第 1 のモデルをハードウェア記述言語

により記述された第2のモデルに変換するソフトウェア・ハードウェア言語モデル変換方法であって、

同一の共有変数へ書き込みを行う複数の並列手続きが前記第1のモデルに含まれているか否かを考慮せずに、前記第1のモデルを前記第2のモデルに変換し、

これにより得られた前記第2のモデルに、前記同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出し、

これにより前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、前記共有変数の値を保持する信号へ出力する値解決プロセスを生成することを特徴とするソフトウェア・ハードウェア言語モデル変換方法。

【請求項11】

ソフトウェア記述言語により記述された第1のモデルをハードウェア記述言語により記述された第2のモデルに変換するソフトウェア・ハードウェア言語モデル変換方法であって、

同一の手続きを呼び出す複数の並列手続きが前記第1のモデルに含まれているか否かを考慮せずに、前記第1のモデルを前記第2のモデルに変換し、

これにより得られた前記第2のモデルに、前記同一の手続きを呼び出す複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出し、

これにより前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する手続き呼び出し解決プロセスを生成することを特徴とするソフトウェア・ハードウェア言語モデル変換方法。

【請求項12】

ソフトウェア記述言語により記述された第1のモデルをハードウェア記述言語により記述された第2のモデルに変換するソフトウェア・ハードウェア言語モデ

ル変換装置としてコンピュータを機能させるためのプログラムを記録したコンピュータ読取り可能な記録媒体であって、

同一の共有変数へ書き込みを行う複数の並列手続きが前記第1のモデルに含まれているか否かを考慮せずに、前記第1のモデルを前記第2のモデルに変換する変換機能と、

この変換機能により得られた前記第2のモデルに、前記同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検出機能と、

前記検出機能により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、前記共有変数の値を保持する信号へ出力する値解決プロセスを生成する値解決プロセス生成機能とをコンピュータに実現させるためのプログラムを記録したコンピュータ読取り可能な記録媒体。

【請求項13】

ソフトウェア記述言語により記述された第1のモデルをハードウェア記述言語により記述された第2のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置としてコンピュータを機能させるためのプログラムを記録したコンピュータ読取り可能な記録媒体であって、

同一の手続きを呼び出す複数の並列手続きが前記第1のモデルに含まれているか否かを考慮せずに、前記第1のモデルを前記第2のモデルに変換する変換機能と、

この変換機能により得られた前記第2のモデルに、前記同一の手続きを呼び出す複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検出機能と、

前記検出機能により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同

時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する手続き呼び出し解決プロセスを生成する手続き呼び出し解決プロセス生成機能とをコンピュータに実現させるためのプログラムを記録したコンピュータ読取り可能な記録媒体。

【請求項 1 4】

ソフトウェア記述言語により記述された第 1 のモデルをハードウェア記述言語により記述された第 2 のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置としてコンピュータを機能させるためのプログラムであって、

同一の共有変数へ書き込みを行う複数の並列手続きが前記第 1 のモデルに含まれているか否かを考慮せずに、前記第 1 のモデルを前記第 2 のモデルに変換する変換機能と、

この変換機能により得られた前記第 2 のモデルに、前記同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検出機能と、

前記検出機能により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、前記共有変数の値を保持する信号へ出力する値解決プロセスを生成する値解決プロセス生成機能とをコンピュータに実現させるためのプログラム。

【請求項 1 5】

ソフトウェア記述言語により記述された第 1 のモデルをハードウェア記述言語により記述された第 2 のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置としてコンピュータを機能させるためのプログラムであって、

同一の手続きを呼び出す複数の並列手続きが前記第 1 のモデルに含まれているか否かを考慮せずに、前記第 1 のモデルを前記第 2 のモデルに変換する変換機能と、

この変換機能により得られた前記第 2 のモデルに、前記同一の手続きを呼び出す複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検

出機能と、

前記検出機能により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する手続き呼び出し解決プロセスを生成する手続き呼び出し解決プロセス生成機能とをコンピュータに実現させるためのプログラム。

【請求項 1 6】

計算に関する仕様および通信に関する仕様を含むシステム仕様モデルを設計するための仕様モデル記述手段と、

システム仕様モデルにおいて、特定のアーキテクチャに対して、システム仕様モデルの内容を保ったまま構造を分配して計算に関する仕様および通信に関する仕様の一部分をアーキテクチャに割り当てるためのアーキテクチャ探索手段と、

システム仕様モデルにおいて、割り当てられた特定のアーキテクチャ上の仕様要素間の通信手順を合成するためのコミュニケーション合成手段と、

システム仕様モデルからハードウェア仕様モデルを生成するためのハードウェア仕様生成手段と、

システム仕様モデルからソフトウェア仕様モデルを生成するためのソフトウェア仕様生成手段とを備え、

前記ハードウェア仕様生成手段は、請求項 1 ないし 9 のいずれか 1 項に記載のソフトウェア・ハードウェア言語モデル変換装置に相当する機能を含むものであることを特徴とするシステム設計支援装置。

【請求項 1 7】

前記仕様モデル記述手段、前記アーキテクチャ探索手段または前記コミュニケーション合成手段の少なくとも一つにおいて、システム仕様モデルを部品化し、これを設計に再利用するための部品化・再利用手段を更に備えたことを特徴とする請求項 1 6 に記載のシステム設計支援装置。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、ソフトウェア記述をハードウェア動作記述に自動変換するソフトウェア・ハードウェア言語モデル変換装置及び方法並びにシステム設計支援装置に関する。

【0 0 0 2】

【従来の技術】

昨今、CやC++さらには仕様記述言語 *SpecC* (“*SpecC: Specification Language and Methodology*”, Daniel D. Gajski, Kluwer Academic Publishers, Dordrecht, ISBN0-7923-7822-9に詳しい)といったソフトウェア記述言語で、システム全体あるいはハードウェアを設計することが注目されている。すなわち、ソフトウェア記述言語により仕様を記述し、これを、ハードウェアの詳細設計に移行する際に下流の論理合成ツールなどに接続するため、*VHDL*や*Verilog*等のハードウェア記述言語による仕様記述に変換するものである。

【0 0 0 3】

C言語からハードウェア言語モデルを生成するツールは、すでに製品化されているものもある。しかし、そのときに入力するコードはシーケンシャルな処理を記述したものに限られており、部分的な設計においては役に立つものの、より大規模な設計を行おうとすると、シーケンシャルな仕様では書ききれない部分があるので不都合がある。従って、並列プログラムをサポートできるソフトウェア・ハードウェア言語モデル変換が望まれている。

【0 0 0 4】

【発明が解決しようとする課題】

ソフトウェア記述言語による記述で並列プログラムを含むとき、その記述を動作レベルのハードウェア記述言語に変換する場合、共有変数の取り扱いを素直に変換することができないという問題点があった。つまり、ソフトウェアでは代入が実行される順序に従って値が決まるが、ハードウェア記述においては各並列プログラムからの信号出力を単純に結線すると値が不定または代入順序とは関係なく決まる値となってしまう。そこで、システムティックに共有変数への代入処理

をハードウェア記述へ変換を行う方法が望まれる。

【0005】

また、ソフトウェア記述言語による記述で並列に動作し得る手続きの呼び出しを含むとき、その記述を動作レベルのハードウェア記述言語に変換する場合、同一の手続きに対する呼び出しの取り扱いを素直に変換することができないという問題点があった。つまり、並列手続きをプロセスへ置きかえた場合、同時に同じ手続きが呼び出され得るが、素直にプロセス間の信号によるシェークハンドで表すと、排他的呼び出しにならないことになってしまう。そこで、システムティックに排他的呼び出しの機構をコード生成する方法が望まれる。

【0006】

本発明は、上記事情を考慮してなされたもので、並列プログラムとその間に共有する変数を含むソフトウェア記述を同様の動作をする動作レベルのハードウェア記述に変換することのできるソフトウェア・ハードウェア言語モデル変換装置及び方法並びにシステム設計支援装置を提供することを目的とする。

【0007】

また、本発明は、ソフトウェア言語による並列プログラムをハードウェア記述に変換する際、手続き呼び出しに相当する排他的なプロセス実行処理を実現することのできるソフトウェア・ハードウェア言語モデル変換装置及び方法並びにシステム設計支援装置を提供することを目的とする。

【0008】

【課題を解決するための手段】

本発明は、ソフトウェア記述言語により記述された第1のモデルをハードウェア記述言語により記述された第2のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置であって、同一の共有変数へ書き込みを行う複数の並列手続きが前記第1のモデルに含まれているか否かを考慮せずに、前記第1のモデルを前記第2のモデルに変換する変換手段と、この変換手段により得られた前記第2のモデルに、前記同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検出手段と、前記検出手段により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全

部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、前記共有変数の値を保持する信号へ出力する値解決プロセスを生成する値解決プロセス生成手段とを備えたことを特徴とする。

【0009】

これによって、並列プログラムとその間に共有する変数を含むソフトウェア記述を同様の動作をする動作レベルのハードウェア記述に変換することができるようになる。

【0010】

好ましくは、前記値解決プロセス生成手段により生成された前記値解決プロセスのうち、それが対象とするプロセスが、同一の手続きを呼び出す複数の並列手続きに相当する複数の並列プロセスであるものについては、該値解決プロセスを、各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する（例えば、予め定められた順番で実行する）手続き呼び出し解決プロセスに変換するプロセス変換手段を更に備えるようにしてもよい。

【0011】

これによって、ソフトウェア言語による並列プログラムをハードウェア記述に変換する際、手続き呼び出しに相当する排他的なプロセス実行処理を実現することができるようになる。

【0012】

また、本発明は、ソフトウェア記述言語により記述された第1のモデルをハードウェア記述言語により記述された第2のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置であって、同一の手続きを呼び出す複数の並列手続きが前記第1のモデルに含まれているか否かを考慮せずに、前記第1のモデルを前記第2のモデルに変換する変換手段と、この変換手段により得られた前記第2のモデルに、前記同一の手続きを呼び出す複数の並列手続きに相当する複数の並

列プロセスが存在するか否か検出する検出手段と、前記検出手段により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する（例えば、予め定められた順番で実行する）手続き呼び出し解決プロセスを生成する手続き呼び出し解決プロセス生成手段とを備えたことを特徴とする。

【0013】

これによって、ソフトウェア言語による並列プログラムをハードウェア記述に変換する際、手続き呼び出しに相当する排他的なプロセス実行処理を実現することができるようになる。

【0014】

また、本発明は、ソフトウェア記述言語により記述された第1のモデルをハードウェア記述言語により記述された第2のモデルに変換するソフトウェア・ハードウェア言語モデル変換方法であって、同一の共有変数へ書き込みを行う複数の並列手続きが前記第1のモデルに含まれているか否かを考慮せずに、前記第1のモデルを前記第2のモデルに変換し、これにより得られた前記第2のモデルに、前記同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出し、これにより前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、前記共有変数の値を保持する信号へ出力する値解決プロセスを生成することを特徴とする。

【0015】

また、本発明は、ソフトウェア記述言語により記述された第1のモデルをハードウェア記述言語により記述された第2のモデルに変換するソフトウェア・ハードウェア言語モデル変換方法であって、同一の手続きを呼び出す複数の並列手続きが前記第1のモデルに含まれているか否かを考慮せずに、前記第1のモデルを

前記第2のモデルに変換し、これにより得られた前記第2のモデルに、前記同一の手続きを呼び出す複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出し、これにより前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する手続き呼び出し解決プロセスを生成することを特徴とする。

【0016】

また、本発明に係るシステム設計支援装置は、計算に関する仕様および通信に関する仕様を含むシステム仕様モデルを設計するための仕様モデル記述手段と、特定のアーキテクチャに対しシステム仕様モデルの内容を保ったまま当該システム仕様モデルの持つ構造を分配して、当該システム仕様モデルに含まれる計算に関する仕様および通信に関する仕様の所定部分を当該特定のアーキテクチャに割り当てるためのアーキテクチャ探索手段と、特定のアーキテクチャに対し仕様要素が割り当てられたシステム仕様モデルにつき、当該割り当てられた特定のアーキテクチャ上の仕様要素間の通信手順を合成するためのコミュニケーション合成手段と、システム仕様モデルをもとにハードウェア仕様モデルを生成するためのハードウェア仕様生成手段と、システム仕様モデルをもとにソフトウェア仕様モデルを生成するためのソフトウェア仕様生成手段とを備え、前記ハードウェア仕様生成手段は、請求項1ないし9のいずれか1項に記載のソフトウェア・ハードウェア言語モデル変換装置に相当する機能を含むものであることを特徴とする。

【0017】

なお、装置に係る本発明は方法に係る発明としても成立し、方法に係る本発明は装置に係る発明としても成立する。

また、装置または方法に係る本発明は、コンピュータに当該発明に相当する手順を実行させるための（あるいはコンピュータを当該発明に相当する手段として機能させるための、あるいはコンピュータに当該発明に相当する機能を実現させ

るための) プログラムとしても成立し、該プログラムを記録したコンピュータ読取り可能な記録媒体としても成立する。

【0018】

【発明の実施の形態】

以下、図面を参照しながら発明の実施の形態を説明する。

【0019】

図18に、本発明の実施の形態に係るシステム設計支援装置100の構成例を示す。図18に示されるように、システム設計支援装置100は、仕様モデル記述部101、アーキテクチャ探索部102、コミュニケーション合成部103、ハードウェア仕様生成部104、部品化・再利用部105、ソフトウェア仕様生成部106、システム仕様記録部107を備えている。システム設計支援装置100は、例えば、コンピュータを用いて構成可能であり、この場合、各処理部分はプログラムによって実施できる。

【0020】

システム仕様記録部107は、入力データや各処理部の処理結果の全部又は一部、例えば、計算機で実行されるソフトウェアの仕様や、半導体を組み合わせたハードウェアの仕様や、ソフトウェアとハードウェアを組み合わせた組み込みシステムの仕様や、ワークフロー等のビジネスプロセスの仕様に対しシステムレベルでの仕様等や、その他の必要なデータを記録するためのものである。

【0021】

仕様モデル記述部101は、設計者が予め定められた仕様記述形式に従って計算内容の仕様、通信内容の仕様を記述することを支援するためのものである。仕様モデル記述部101の支援の結果、仕様モデルが生成される。仕様記述形式には、例えば、構造化プログラミング言語に代表される構造化テキスト形式や、グラフを利用する構造図形式や、テーブルを利用する表形式が有る。

【0022】

アーキテクチャ探索部102は、与えられた仕様記述モデルと、アーキテクチャ(例えば、ハードウェア実行環境とソフトウェア実行環境の構成)に対し、仕様の内容を保存したまま仕様モデルの部分構造を分割しアーキテクチャ要素に分

配する。すなわち、仕様モデル記述部 1 0 1 で設計された計算内容の仕様や通信内容の仕様を構成する部品をアーキテクチャ要素に割り当てる。

【 0 0 2 3 】

コミュニケーション合成部 1 0 3 は、アーキテクチャ探索部 1 0 2 で分配された通信仕様要素間に対して通信手順（プロトコル）を挿入し、通信内容の仕様が挿入された通信手順と整合性を保つためのプロトコル変換（通信手順の組替え）を行う。

【 0 0 2 4 】

なお、アーキテクチャ探索部 1 0 2 が処理対象とする仕様記述モデルは、仕様モデル記述部 1 0 1 を利用して記述されたものに限定されず、他のツール等で記述されたものでもよい。また、コミュニケーション合成部 1 0 3 が処理対象とする仕様記述モデルは、アーキテクチャ探索部 1 0 2 の出力結果（またはこの結果に修正を加えたもの）に限定されず、他のツール等で記述されたものでもよい。

【 0 0 2 5 】

ハードウェア仕様生成部 1 0 4 は、システム仕様からハードウェア記述言語などによるハードウェア仕様を生成する。

【 0 0 2 6 】

ソフトウェア仕様生成部 1 0 6 は、システム仕様からソフトウェア記述言語などによるソフトウェア仕様を生成する。

【 0 0 2 7 】

ハードウェア仕様生成部 1 0 4 やソフトウェア仕様生成部 1 0 6 は、仕様モデル記述部 1 0 1 の出力結果（またはこの結果に修正を加えたもの）、アーキテクチャ探索部 1 0 2 の出力結果（またはこの結果に修正を加えたもの）、コミュニケーション合成部 1 0 3 の出力結果（またはこの結果に修正を加えたもの）、あるいは他のツール等で記述されたものなどを処理対象とすることができる。

【 0 0 2 8 】

部品化・再利用部 1 0 5 は、仕様モデル記述部 1 0 1、アーキテクチャ探索部 1 0 2、コミュニケーション合成部 1 0 3 の全部または一部の設計段階において、仕様を部品化し、これを設計に再利用するためのものである。

【 0 0 2 9 】

以下詳述するソフトウェア・ハードウェア言語モデル変換装置は、図 1 8 のシステム設計支援装置のハードウェア仕様生成部 1 0 4 に用いられるもので、システム仕様から HDL 記述を生成することを可能にするものである。

【 0 0 3 0 】

図 1 に、本発明の一実施形態に係るソフトウェア・ハードウェア言語モデル変換装置（以下、言語モデル変換装置と呼ぶ）の構成例を示す。

【 0 0 3 1 】

図 1 に示されるように、本言語モデル変換装置は、骨格コード生成部 1、コード置換部 2、値解決プロセス生成部 3 を備えている。

【 0 0 3 2 】

本言語モデル変換装置は、SpecC 言語等のソフトウェア記述言語により記述されたモデルから、VHDL 等のハードウェア記述言語により記述された（同様の動作をする）モデルを生成するものである。

【 0 0 3 3 】

まず、図 2（ソフトウェア記述言語モデルの説明図）および図 3（ハードウェア記述言語の説明図）を参照して、本言語モデル変換装置の概要について説明する。なお、図 2 および図 3 において、1 0 2 1 は共有変数、1 0 2 3 は並列に動作し得る手続き、1 0 2 2 は手続きからの出力、1 0 1 1 は共有変数の値を保持する信号、1 0 1 5 は互いに並行に実行されるプロセス、1 0 1 2 は本言語モデル変換装置が生成する値解決プロセス（矢印が信号の入出力を表す）、1 0 1 3 はプロセスの代入データ信号、1 0 1 4 はプロセスの代入タイミング信号である。

【 0 0 3 4 】

図 2 のようにソフトウェア記述言語において並列プログラム間の共有変数がある場合、本言語モデル変換装置は、図 3 のようにハードウェア記述言語において、各共有変数に対して、値保持信号（1 0 1 1）とその共有変数に書き込みアクセスし得る各並列プログラム（1 0 2 3）に対応するプロセス（1 0 1 5）ごとに、代入データ信号（1 0 1 3）と代入タイミング信号（1 0 1 4）を生成し、

代入タイミング信号（1014）の変化に従って、値保持信号（1011）への出力を、代入タイミング信号（1014）と組になるデータ信号（1013）に切り替える、値解決プロセス（1012）を生成する。値解決プロセス（1012）の生成は、入力コードにおけるその共有変数代入の処理に対応する部分にデータ信号（1013）への代入と代入タイミング信号（1014）の変化を与える処理（図8参照）を生成することによって行う。

【0035】

また、ここでは、図4のように入力されるソフトウェア記述言語モデルにおいて手続きが階層化されていて出力されるハードウェア記述言語モデルにおいて並列プロセスが階層化されている場合、図5のように値解決プロセスは階層ごとにあって信号がその階層に従って接続される。その場合、下位階層の値解決プロセスの出力には、代入タイミング信号が加わる。階層があるときは、値反映の待ちを挿入する。

【0036】

以下、本言語モデル変換装置の処理について説明する。

【0037】

図6に、本言語モデル変換装置の処理手順の一例を示す。図6に示されるように本手順は、骨格コード生成処理（処理1～6）と、コード置換生成処理（処理7，8）と、値解決プロセス生成処理（処理9，10）とを行うものである。

【0038】

なお、後述する処理1～10は、図7に示すような実行順序制約（矢印の終点側にある処理はその矢印の起点側にある全処理の終了後に実行する）を満たす限り、どのような順番で処理を行っても構わない。また、複数の処理を並列に行っても良い。

【0039】

ここでは、SpecC言語のソースコードを入力して、VHDLソースコードを生成する場合を例にとって説明する。また、出力するVHDLコードは、CLKという信号を各EntityでIN属性の入力ポート信号として持ち、その入力値は同期クロックであると仮定しているものとして説明する。

【0040】

＜骨格コード生成処理（骨格コード生成部1）＞

（処理1）SpecC言語の仕様は、C言語のグローバル関数・グローバル変数とclassと呼ばれる要素からなる。本言語モデル変換装置においては、classと呼ばれる要素のみを変換するものとする。classは、behaviorとchannelの2種類ある。本言語モデル変換装置では、classはいずれもVHDLのEntityとArchitectureの組に対応させる。

【0041】

（処理2）SpecC言語のclassは、port変数とローカル変数、そして、関数からなる。本言語モデル変換装置では、port変数は対応するArchitectureのport信号に、ローカル変数はArchitectureのローカル信号に、関数はArchitectureのprocessにそれぞれ対応させる。

【0042】

（処理3）SpecC言語のclassにおけるport変数には、入力属性（in）、出力属性（out）、入出力属性（inout）の3種類の方向属性がある。上記の処理2において、port変数のport信号への置き換えを、方向属性によって、次のように行う。

- ・入力属性（in）の場合、IN属性の信号（合計1つ）を対応させる。
- ・出力属性（out）の場合、OUT属性の信号（合計1つ）を対応させる。

・入出力属性（inout）の場合、IN属性の信号とOUT属性の信号（合計2つ）を対応させる。

【0043】

（処理4）SpecC言語のclass内の関数に対し、呼び出し用の信号として、実行トリガー信号（processにとっての入力信号）、実行状態信号（processにとっての出力信号）を定義する。classの外から呼ばれる関数に関してはEntityのport信号として定義する。関数に引数や戻

り値があれば、それに対応する入出力信号を同様に `port` に定義する。

【0044】

(処理5) `process` の冒頭において、外部からの関数の呼び出しに対応するコードを次のように生成する。

```
process
begin
    実行状態信号<= '0' ;
    wait until 実行トリガー信号= '1' ;
    実行状態信号<= '1' ;
    wait until CLK= '1' and CLK' EVENT;
    ---関数内の処理
end process
```

(処理6) 関数内のコードについて、関数内のローカル変数はVHDLの `process` 変数に対応させる。変数の代入文は代入文に、`wait` は `wait` 文に対応させる。`if` 等の制御構文はVHDLの `if` 等に対応させる。関数の呼び出しは、上記の処理4で生成した信号を用いて次のようなコードに対応させる。

```
実行トリガー信号<= '1' ;
wait until 実行状態信号= '1' ;
実行トリガー信号<= '0' ;
wait until 実行状態信号= '0' ;
```

<コード置換処理(コード置換部2)>

(処理7) VHDLの `process` ブロック内部における信号代入は、その場で即座に信号値に反映されるわけではない。反映されるのは、`wait` 文が処理されるときである。その効果を考慮して、次のように変換する。VHDLの信号に対して、一時 `process` 変数に対応して定義する。`wait` 文の直後に、変数:=信号;の代入文を作り、`wait` 文の直前に、信号<=変数;の代入文を作る。SpecC言語の関数外変数への代入文は一時変数への代入文とする

。また、値の参照も一時変数の値の参照とする。

【0045】

一例として、a を関数外変数として、次のような SpecC 言語コードを考える。

```
waitfor 100;
a=1;
a=a+1;
waitfor 100;
```

これに対応する VHDL コードは次のようになる。a が SpecC 変数 a に対応する信号であり、tmp_a がその信号に対応する一時 process 変数である。

```
wait for 100 ns;
tmp_a:=a;
tmp_a:=1;
tmp_a:=tmp_a+1;
a<=tmp_a;
wait for 100 ns;
```

ただし、代入先が SpecC 言語の port 変数に対応する場合、代わりに OUT 属性の信号に代入する。ローカル信号の場合は、特に変えなくて良い。

【0046】

SpecC 言語の変数の型が event であった場合、特に一時変数は作らない。

【0047】

イベントが起こったことを表す方法は、イベント信号が 1 クロック間 '1' の値をとることとする。イベント変数に関しては notify e; が代入に相当する。

```

notify e;
waitfor 100;

```

は、次のようなコードに対応させる。

```

e<= '1' ;
wait until CLK= '1' and CLK' EVENT ;
c<= '0' ;
wait for 100 ns;

```

イベントの参照は、wait e ;で行われるが、このコードに対応するVHDLコードは、

```
wait e= '1' ;
```

となる。

【0048】

(処理8) SpecC言語のclassは他のクラスをインスタンス化することができる。インスタンス化はVHDLのport__map宣言に対応させる。ただし、portが上記の処理3で複数に分裂している場合があるので、ローカル変数に接続している場合は、同じ信号を、ポート変数を接続している場合は、同じ方向属性の信号を接続する。

【0049】

例えば、次のようなSpecC言語コードを考える。

```

behavior B0(inout int b);
behavior B1(inout int a){
    B2 aB2(a);
    // ...
};

```

これに対応するVHDLコードは次のようになる。


```

Entity B1 is port(a:in integer; a_out: out integer; ... );end B1;
Architecture B1 od B1 is
  Component B0 is port(b: in integer; b_out: out integer; ... );end c
omponent;

  begin

    aB2: B2 port map(a,a_out,...);

    -- ...

  end B1;

```

<値解決プロセス生成処理（値解決プロセス生成部3）>

（処理9）各Architecture内で、複数のprocessやport_mapからの代入が起こっている信号を検出する。

検出した信号sigに対して、（a）代入しようとするプロセスfに対してデータ信号sig_out_fと代入タイミング信号sig_otrig_fをArchitectureローカル信号として定義する。

（b）（a）で生成した信号を入力とし、信号sigを出力とする値解決プロセスを生成する。

プロセスの実装するアルゴリズムは、図8のようである。

（c）各プロセスのsigへの代入文sig<=式；を次のように置き換える

```

sig_out_f<=式;
sig_otrig<= '1' ;
wait until CLK= '1' and CLK' EVENT;
sig_otrig<= '0' ;

```

関数実行制御用の信号の値解決プロセスに関しては、実行トリガーと引数に関する処理をのをひとまとめで行う。実行トリガー信号は代入タイミング信号と同様な役目を負わせることができるからである。

【0050】

(処理10) 上記の処理9で生成した値解決プロセスの出力先の信号が Entity Port の場合、処理9で、代入タイミング信号を生成している場合がある。その場合は、値解決プロセスの出力として、代入タイミング信号を追加し、値解決プロセスが処理して、出力先にデータを出力するタイミングで、代入タイミング信号の値を変化させるような処理を追加する。

【0051】

以上の手順によって、例えば図4のような構造の SpecC 言語コードから図5のような構造の VHDL コードが生成される。

【0052】

次に、本言語モデル変換装置の他の処理例について説明する。

【0053】

図9に、本言語モデル変換装置の処理手順の他の例を示す。

【0054】

SpecC 言語コードは一般に class の間に木構造が形成されている。従って、SpecC 言語コードを入力して、図6の手順を行って出力した VHDL コードの Entity-Architecture 組間にも同様な木構造が形成される。すなわち、図6の手順の値解決プロセス生成処理(処理9、処理10)で生成した値解決プロセスについて、一般にその木構造に沿って値解決プロセスの木構造ができる。

【0055】

図9の手順では、まず図6と同様の手順を行い、その後に、値解決プロセス生成部3で後述する値解決プロセス除去処理(処理11、12)を行って、プログラムが階層化されている場合に値解決プロセスを1つにするものである。例えば、図6の手順で作られた値解決プロセス(図5参照)のうち最も上位にある値解決プロセスに、階層化されている値解決プロセスを統合する。この結果、例えば、図10のようになる。値解決プロセス除去処理では、階層の最も上に位置するものだけを残して後は除去するために、下位層に含まれる代入データ信号と代入タイミング信号を各コンポーネントのポート信号にして上位のコンポーネントに接続する。

【 0 0 5 6 】

以下では、値解決プロセス除去処理について説明する。

【 0 0 5 7 】

＜値解決プロセス除去処理（値解決プロセス生成部 3）＞

（処理 1 1）出力に代入タイミング信号がない値解決プロセスが、値解決プロセスの階層の頂上にある。それ以外を「除去可能な値解決プロセス」と呼ぶものとする。除去可能な値解決プロセスに関して、そのプロセスを除去する。

【 0 0 5 8 】

（処理 1 2）上記の処理 1 1 で除去されたプロセスの入力を Entity ポートに展開する。除去されたプロセスの出力ポートは除去する。これを値解決プロセスの下の階層から順に行う。その中間階層の値解決プロセスを処理している場合、port_map の入力信号ポート部が別のポート群に置き換わっているがその置き換わった信号群を入力信号としてみなして処理を行う。最終的に頂上の値解決プロセスの入力は、全ての階層のその信号への代入処理を持つプロセスからのデータ信号と代入タイミング信号になる。

【 0 0 5 9 】

以上説明してきたように、本実施形態によれば、並列プログラムとその間に共有する変数を含むソフトウェア記述を同様の動作をする動作レベルのハードウェア記述に変換することができる。

【 0 0 6 0 】

また、並列プロセスが階層化されている場合にも対応できる。なお、値解決プロセスを階層構造にする構成と、最上位の値解決プロセスのみ残す構成とのいずれかを選択できるようにしてもよい（前者の場合は、処理 1 0 まで行い、後者の場合は、処理 1 2 まで行えばよい）。選択の基準としては、例えば、ポート信号の数を減らしたいときは前者の構成を選択し、階層が深く余分な待ち処理を入れたくない場合は後者の構成を選択するようにする。

【 0 0 6 1 】

図 1 1 に、本発明の一実施形態に係る言語モデル変換装置と呼ぶの他の構成例を示す。この言語モデル変換装置は、図 1 の構成に値解決プロセス変換部 4 を付

加したものである。

【0062】

まず、図12（ハードウェア記述言語モデルの説明図）を参照して、値解決プロセス変換処理の概要について説明する。なお、図12において、2015は呼び出す側のプロセス、2016は呼び出される側のプロセス、2010は（呼び出される側の）プロセスへの実行起動トリガー信号、2011は（呼び出す側の）プロセスからの実行状態信号、2012はプロセスを呼び出すための手続き呼出切り替えプロセス（値解決プロセス変換処理により値解決プロセスが変換されたもの）、2013は（呼び出す側の）プロセスから他のプロセスを呼び出すための実行起動トリガー信号、2014は（呼び出す側の）プロセスから他のプロセスを呼び出すときの実行状態信号である。

【0063】

ソフトウェア記述言語モデルにおける並列プログラム間の手続き呼び出しを、ハードウェア記述言語モデルにおいて次のように表現する。

【0064】

並列プログラムから並列プロセスに変換される手続きを呼び出すコードに対して、次のようなコードを生成する。

・手続きを呼び出す際には、呼び出す側のプロセス2015が出力するトリガ信号2013（2011）と、呼び出される側のプロセス2016が出力する実行状態信号2011（2014）を使う。加えて、呼び出す側のプロセスが出力する引数信号と、呼び出される側のプロセスが出力する返り値信号を使うこともある。

（1）トリガは実際に呼び出される側のプロセスが実行されるまで立ち上げておくとする。

（2）呼び出される側のプロセスに対して、（出力）トリガ信号、引数、（入力）実行状態、返り値の信号の切替を行う手続き呼び出し切替プロセスを作る。

（3）呼び出される側のプロセス対する並列プロセスごとに（呼び出しプロセスに対して出力）トリガ信号、引数、（呼び出しプロセスに対して入力）実行状

態、返り値の信号を別に用意する。

【0065】

手続き呼び出し切替プロセス2012は、次のような各入力トリガの立ち上がり条件で実行する処理を逐次に行う（図15参照）。

各入力トリガの立ち上がり条件で実行する処理：

- (1) 他の実行状態信号を下げる。
- (2) 出力トリガを立てる。同時に、（あれば）処理プロセスからの引数信号値を出力引数信号の値とする。
- (3) 入力実行状態信号が立ち上がったら、処理プロセスの実行状態信号を立ち上がらせる。
- (4) 入力実行状態信号が立ち下がると、処理プロセスの状態信号を立ち下がらせる。同時に、（あれば）入力返り値信号値を処理プロセスの返り値信号値にする。

【0066】

また、ここでは、図4のように入力されるソフトウェア記述言語モデルにおいて手続きが階層化されていて出力されるハードウェア記述言語モデルにおいて並列プロセスが階層化されている場合、図14のように各階層ごとに1のプロセスに手続き呼び出し切替プロセスを定義する。ただし、切替える必要のない場合は、手続き呼び出し切替プロセスを介さず結線する。

【0067】

図13に、本言語モデル変換装置の処理手順を示す。図13の手順では、まず図6と同様の手順を行う。ソフトウェア記述言語モデルにおける並列プログラム間の手続き呼び出しも、並列プログラム間の共有変数へのアクセスとして扱われるので、図6と同様の手順によって、並列プログラム間の手続き呼び出しについて、値解決プロセスが生成される。次いで、図13の手順では、値解決プロセス変換部4で後述する値解決プロセス変換処理（処理13）を行って、値解決プロセス（図3参照）のうち並列プログラム間の手続き呼び出しに該当するものを、手続き呼び出し切り替えプロセス（図12参照）に変換する（なお、並列プログラム間の手続き呼び出しに該当しないものは、値解決プロセスとして残る）。

【0068】

以下では、値解決プロセス変換処理について説明する。

【0069】

＜値解決プロセス変換処理（値解決プロセス変換部4）＞

（処理13）骨格コード生成処理の処理4で生成したprocessの制御用信号（実行トリガー信号、実行状態信号、加えて、引数、戻り値データ信号）の制御信号伝達のための値解決プロセスを、値解決プロセス生成処理の処理9で作成したものから取り替える。具体的には次のような置き換えを行う。

【0070】

（処理13-1）

まず、信号の定義の追加を行う。実行状態信号は、各呼び出し側のprocessに対して共通であったものを、呼び出しプロセスごとに用意する。同様に、戻り値の信号も呼び出しprocess毎に追加定義する。

【0071】

（処理13-2）

次に、呼び出し側プロセスにおける信号の置き換えを行う。すなわち、wait untilの条件内の実行状態信号を、処理13-1で生成した信号に置き換える。戻り値信号参照部分に関しても同様に置き換える。

【0072】

（処理13-3）

次に、実行トリガー信号の値解決プロセスを、次のようなものに置き換える。

【0073】

processの入力は、各呼び出しprocessから実行トリガー信号と引数のデータ信号と、加えて、呼び出されるprocessから出力される実行状態信号及び戻り値信号、出力は、各呼び出しprocess信号用の実行状態信号と戻り値信号と、加えて、呼び出されるprocessへの実行トリガー信号及び引数データ信号である。

【0074】

さらに、このprocessは、図15に示すような動作またはこれと等価な

動作を実装する。例えば、実装する動作は、次の通りである。

(1) 実行トリガーを監視する無限ループがある。このループを抜ける条件は、入力の実行トリガー信号の少なくとも1つの値が立ち上がることである。

(2) ループを抜けた直後に、各実行トリガー信号が立ち上がっているという条件の条件分岐を用意する。ただし、この条件分岐は、次のように逐次直列に並んでいるものとする。

```
if 条件1 then
  処理1;
end if;
if 条件2 then
  処理2;
end if;
...
```

(3) 各条件が成り立ったときの処理は、次のようである。

(3-1) 呼び出される process への実行トリガー信号値を立ち上げる。

(3-2) 呼び出される process からの実行状態信号が立ち上がるのを待つ。

(3-3) 呼び出し process への実行状態信号を立ち上げる。

(3-4) 呼び出される process からの実行状態信号が立ち下がるのを待つ。

(3-5) 呼び出し process への実行状態信号が立ち下げ、呼び出される process からの返り値信号値を呼び出し process の返り値信号に伝達する。

【0075】

次に、本言語モデル変換装置の他の処理例について説明する。

【0076】

図16に、本言語モデル変換装置の処理手順の他の例を示す。

【0077】

この手順は、図13の骨格コード生成処理（処理1～6）、コード置換生成処理（処理7、8）、値解決プロセス生成処理（処理9、10）を行い、そして、値解決プロセス除去処理（処理11、12）を行った後に、図13の値解決プロセス変換処理（処理13）を行うものである。

【0078】

すなわち、並列プロセスが階層化されている場合、階層的に生成される手続き呼び出し切替プロセスのうち最も上位に位置するものを残して他の手続き切替プロセスは生成しないようにする。下位のプロセスへの入力出力信号を残したプロセスの入力出力信号とする。そのために必要なポート信号を生成する。

【0079】

この結果、例えば、図14は、図17のようになる。

【0080】

以上説明してきたように、本実施形態によれば、ソフトウェア言語による並列プログラムをハードウェア記述に変換する際、手続き呼び出しに相当する排他的なプロセス実行処理を実現した（排他的呼び出しを可能にした）ハードウェア動作記述に変換することができる。

【0081】

また、並列プロセスが階層化されている場合にも対応できる。なお、手続き呼び出し解決プロセスを階層構造にする構成と、最上位の手続き呼び出し解決プロセスのみ残す構成とのいずれかを選択できるようにしてもよい（前者の場合は、処理10までと処理13を行い、後者の場合は、処理13まで行えばよい）。選択の基準としては、例えば、ポート信号の数を減らしたいときは前者の構成を選択し、階層が深く余分な遅延を入れたくない場合は後者の構成を選択するようにする。

【0082】

なお、以上の各機能は、ソフトウェアとして実現可能である。

また、本実施形態は、コンピュータに所定の手段を実行させるための（あるいはコンピュータを所定の手段として機能させるための、あるいはコンピュータに

所定の機能を実現させるための) プログラムとして実施することもでき、該プログラムを記録したコンピュータ読取り可能な記録媒体として実施することもできる。

【0083】

なお、この発明の実施の形態で例示した構成は一例であって、それ以外の構成を排除する趣旨のものではなく、例示した構成の一部を他のもので置き換えたり、例示した構成の一部を省いたり、例示した構成に別の機能あるいは要素を付加したり、それらを組み合わせたりすることなどによって得られる別の構成も可能である。また、例示した構成と論理的に等価な別の構成、例示した構成と論理的に等価な部分を含む別の構成、例示した構成の要部と論理的に等価な別の構成なども可能である。また、例示した構成と同一もしくは類似の目的を達成する別の構成、例示した構成と同一もしくは類似の効果を奏する別の構成なども可能である。

また、この発明の実施の形態で例示した各種構成部分についての各種バリエーションは、適宜組み合わせて実施することが可能である。

また、この発明の実施の形態は、個別装置としての発明、関連を持つ2以上の装置についての発明、システム全体としての発明、個別装置内部の構成部分についての発明、またはそれらに対応する方法の発明等、種々の観点、段階、概念またはカテゴリに係る発明を包含・内在するものである。

従って、この発明の実施の形態に開示した内容からは、例示した構成に限定されることなく発明を抽出することができるものである。

【0084】

本発明は、上述した実施の形態に限定されるものではなく、その技術的範囲において種々変形して実施することができる。

【0085】

【発明の効果】

本発明によれば、並列プログラムとその間に共有する変数を含むソフトウェア記述を同様の動作をする動作レベルのハードウェア記述に変換することができる。

【0086】

また、本発明によれば、ソフトウェア言語による並列プログラムをハードウェア記述に変換する際、手続き呼び出しに相当する排他的なプロセス実行処理を実現することができる。

【図面の簡単な説明】

【図1】

本発明の実施の形態に係るソフトウェア・ハードウェア言語モデル変換装置の構成例を示す図

【図2】

変数を共有する並列ソフトウェアプログラムのソフトウェア記述の一例について説明するための図

【図3】

変数を共有する並列ソフトウェアプログラムのソフトウェア記述を変換して得られる値解決プロセスを含むハードウェア記述の一例について説明するための図

【図4】

変数を共有する階層化された並列ソフトウェアプログラムのソフトウェア記述の一例について説明するための図

【図5】

変数を共有する階層化された並列ソフトウェアプログラムのソフトウェア記述を変換して得られる階層化された値解決プロセスを含むハードウェア記述の一例について説明するための図

【図6】

同実施形態に係るソフトウェア・ハードウェア言語モデル変換装置の処理手順の一例を示すフローチャート

【図7】

同実施形態に係るソフトウェア・ハードウェア言語モデル変換装置の処理手順の一例を示すフローチャート

【図8】

値解決プロセスによる処理の一例を示すフローチャート

【図9】

同実施形態に係るソフトウェア・ハードウェア言語モデル変換装置の処理手順の他の例を示すフローチャート

【図10】

変数を共有する階層化された並列ソフトウェアプログラムのソフトウェア記述を変換して得られる最上位階層以外の値解決プロセスを削除したハードウェア記述の一例について説明するための図

【図11】

同実施形態に係るソフトウェア・ハードウェア言語モデル変換装置の他の構成例を示す図

【図12】

値解決プロセスを含むハードウェア記述を変換して得られる手続き呼び出し切り替えプロセスを含むハードウェア記述の一例について説明するための図

【図13】

同実施形態に係るソフトウェア・ハードウェア言語モデル変換装置の処理手順のさらに他の例を示すフローチャート

【図14】

階層化された値解決プロセスを含むハードウェア記述を変換して得られる階層化された手続き呼び出し切り替えプロセスを含むハードウェア記述の一例について説明するための図

【図15】

手続き呼び出し切り替えプロセスによる処理の一例を示すフローチャート

【図16】

同実施形態に係るソフトウェア・ハードウェア言語モデル変換装置の処理手順のさらに他の例を示すフローチャート

【図17】

最上位階層以外の値解決プロセスを削除したハードウェア記述を変換して得られる最上位階層以外の手続き呼び出し切り替えプロセスを削除したハードウェア記述の一例について説明するための図

【図18】

同実施形態に係るシステム設計支援装置の構成例を示す

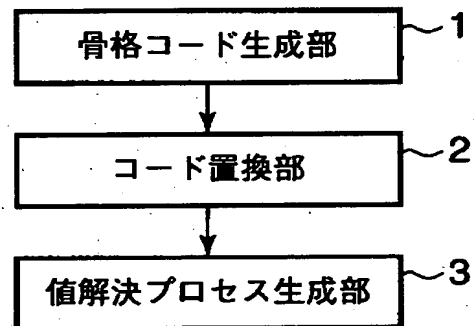
【符号の説明】

- 1 … 骨格コード生成部
- 2 … コード置換部
- 3 … 値解決プロセス生成部
- 4 … 値解決プロセス変換部
- 1 0 0 … システム設計支援装置
- 1 0 1 … 仕様モデル記述部
- 1 0 2 … アーキテクチャ探索部
- 1 0 3 … コミュニケーション合成部
- 1 0 4 … ハードウェア仕様生成部
- 1 0 5 … 部品化・再利用部
- 1 0 6 … ソフトウェア仕様生成部
- 1 0 7 … システム仕様記録部

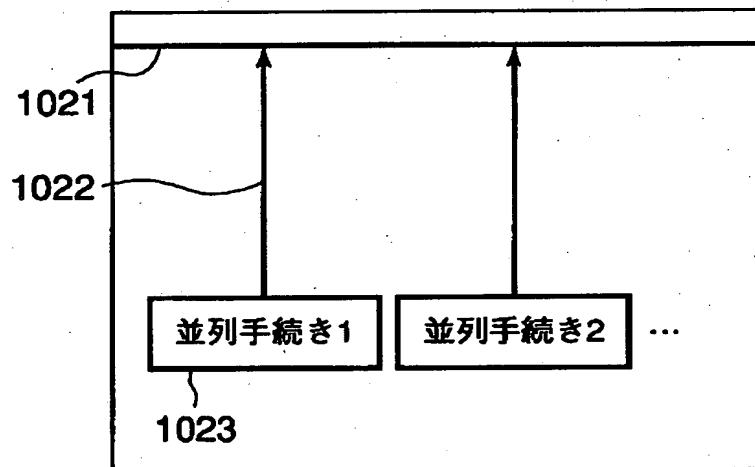
【書類名】

図面

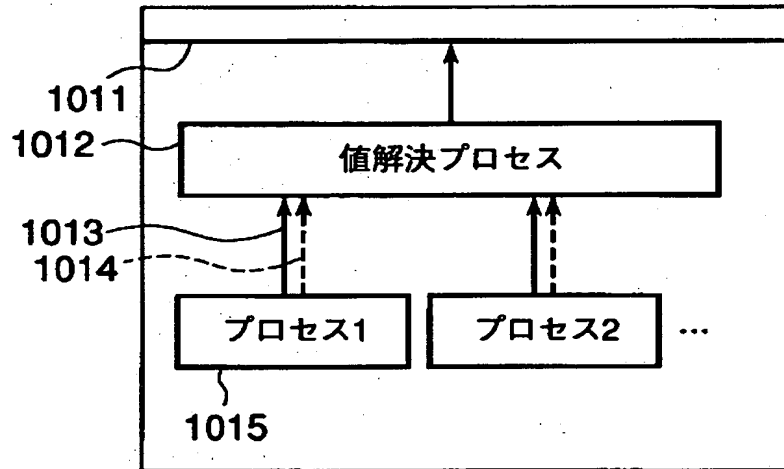
【図 1】



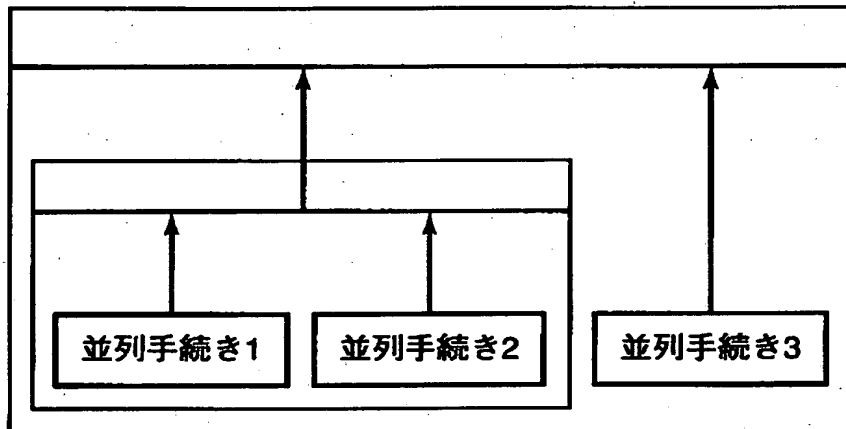
【図 2】



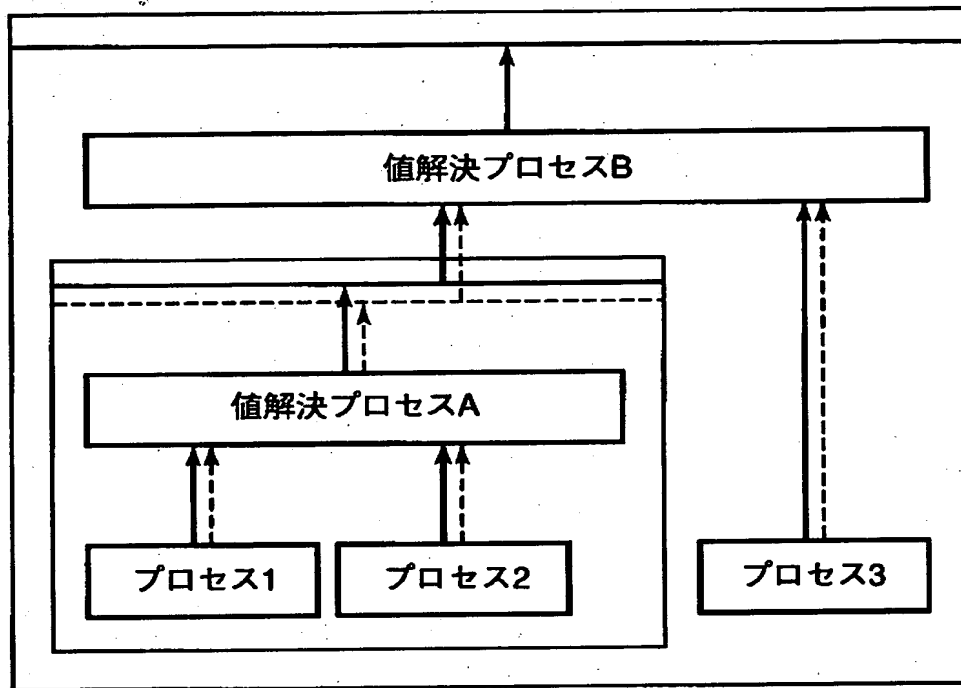
【図3】



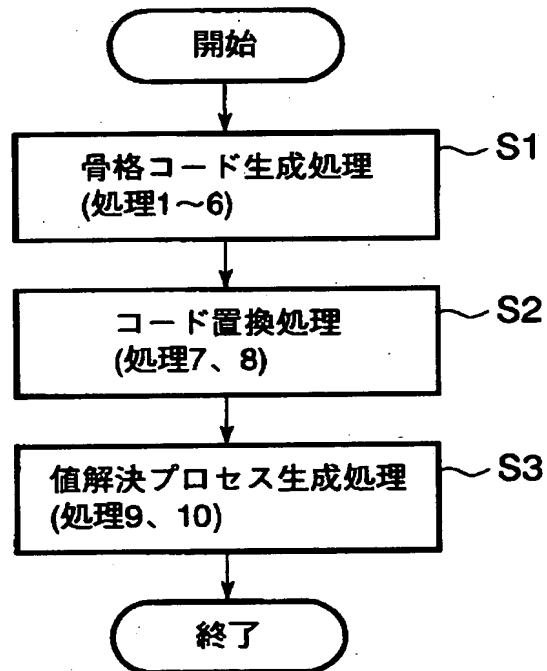
【図4】



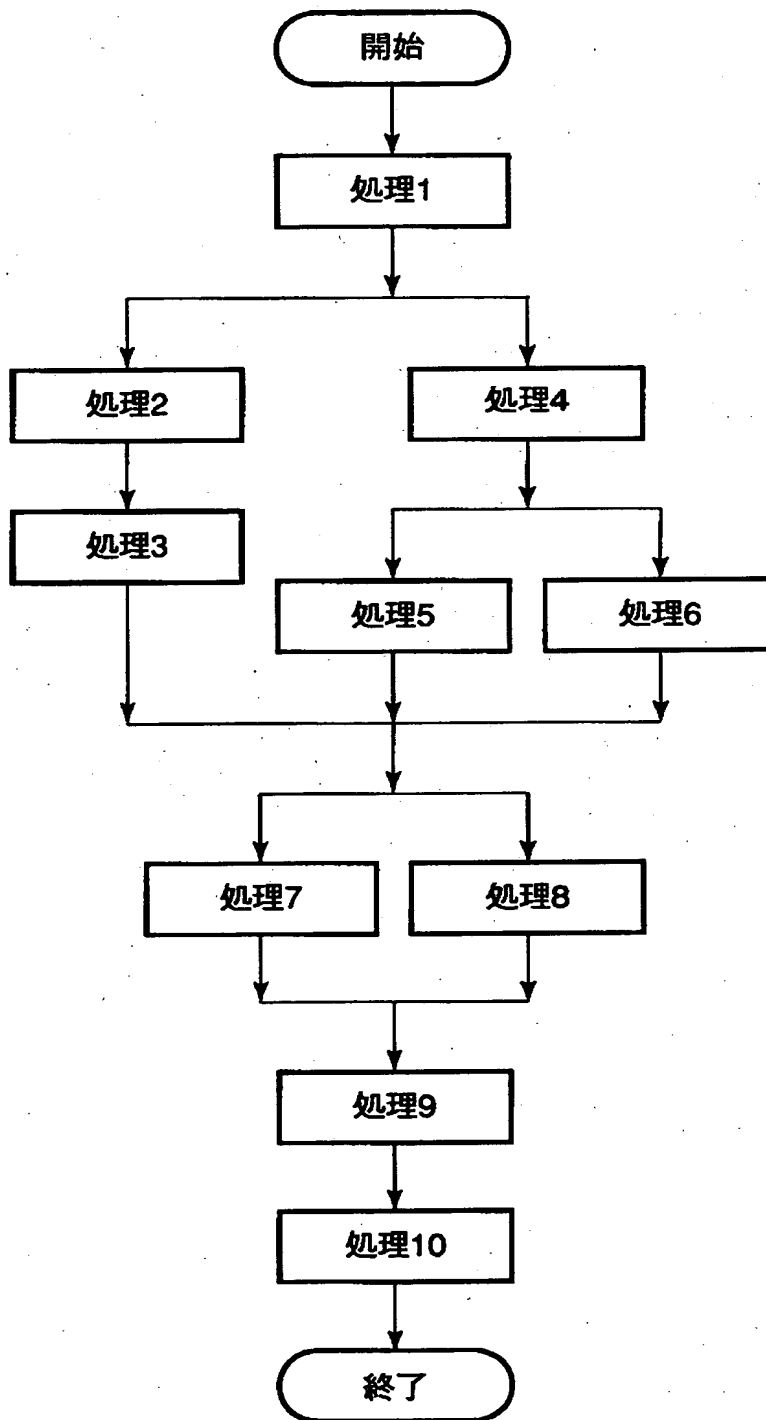
【図5】



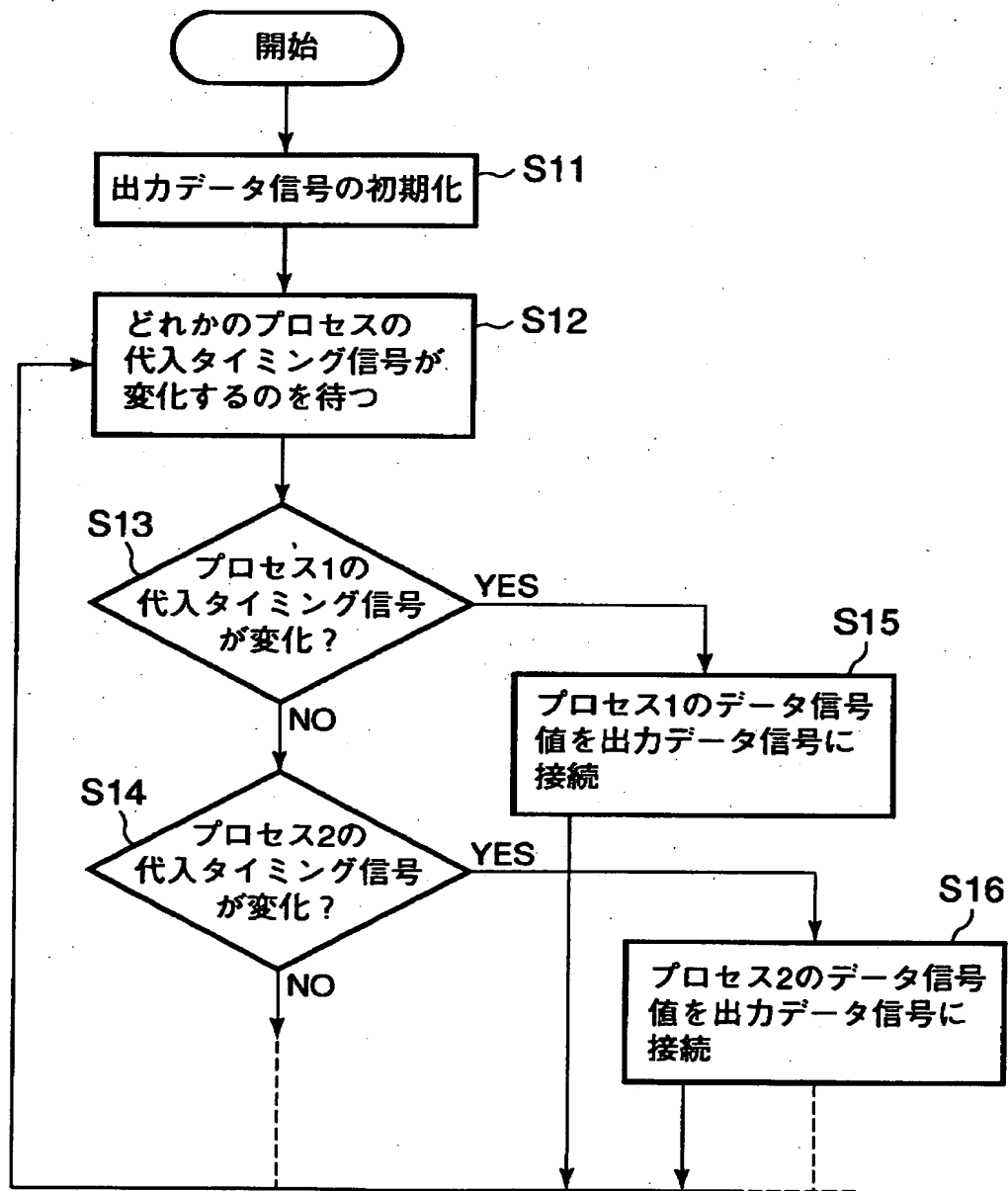
【図6】



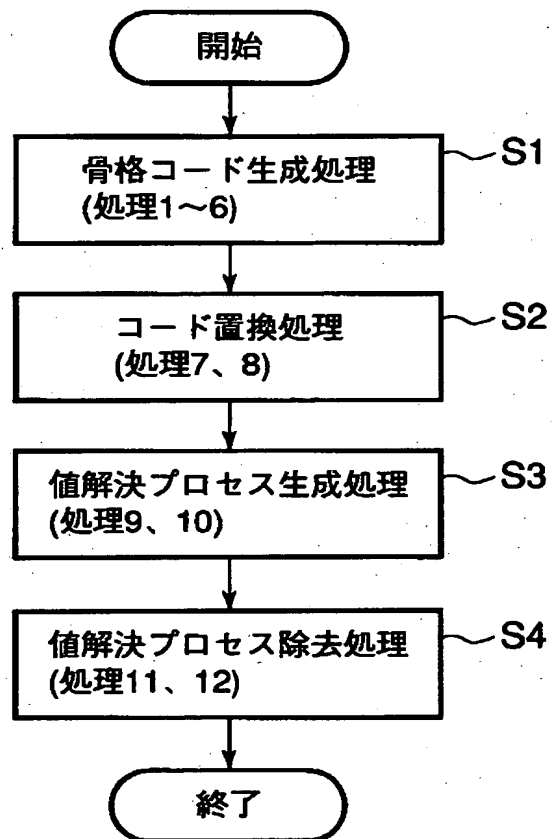
【図 7】



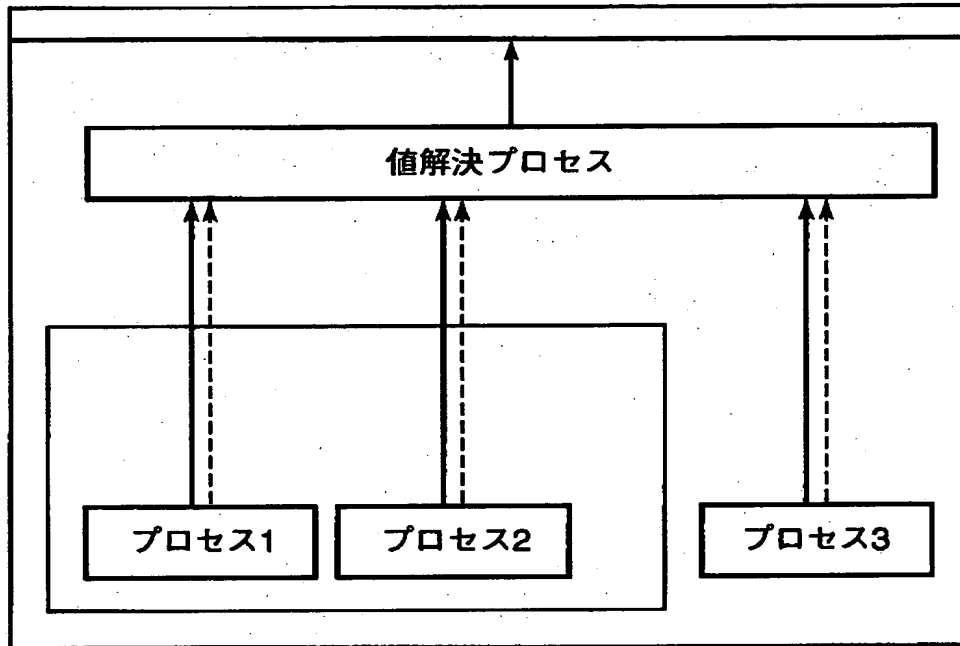
【図 8】



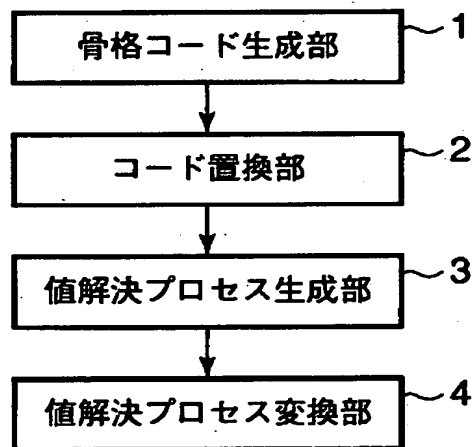
【図 9】



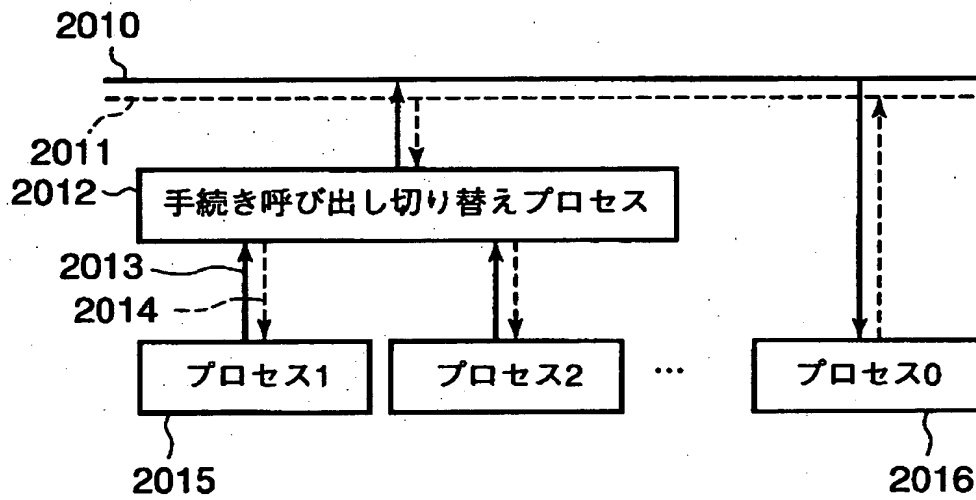
【図10】



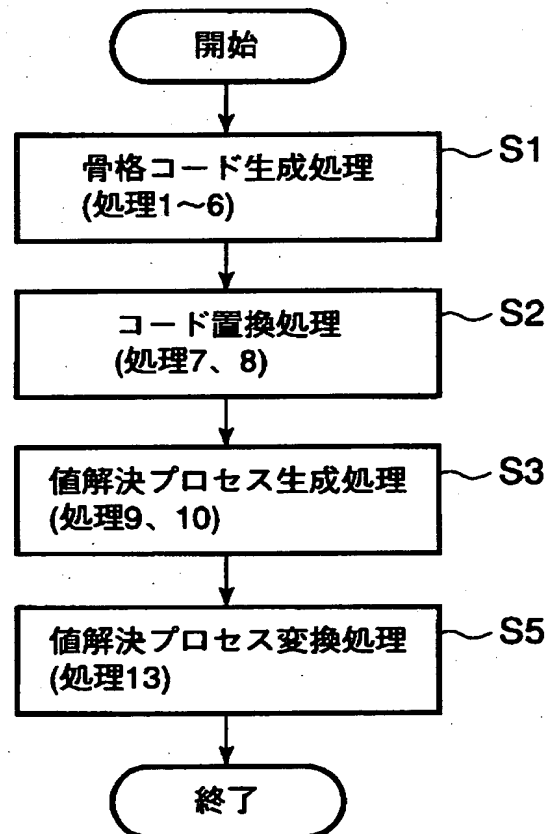
【図11】



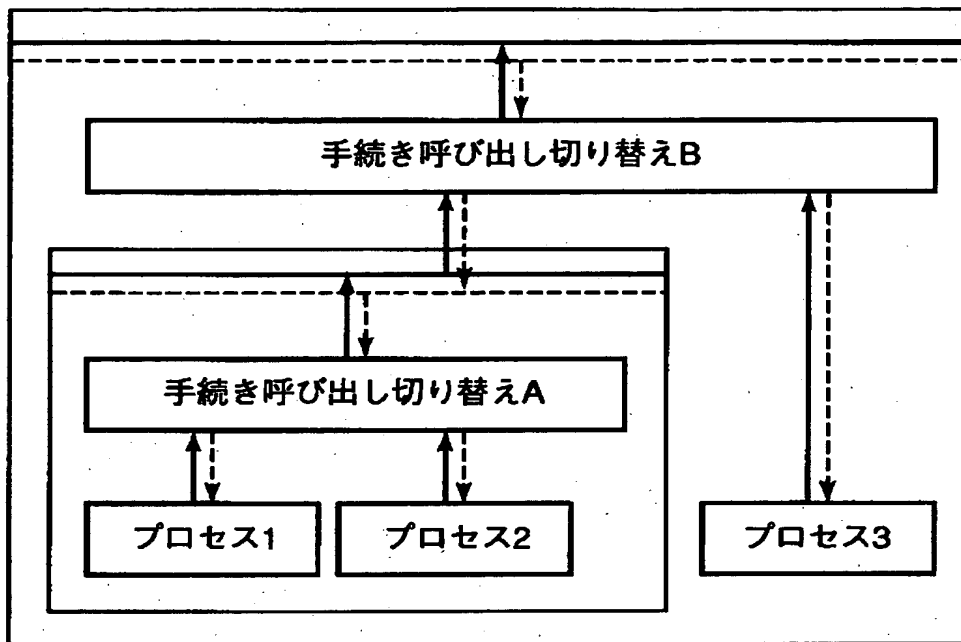
【図 1 2】



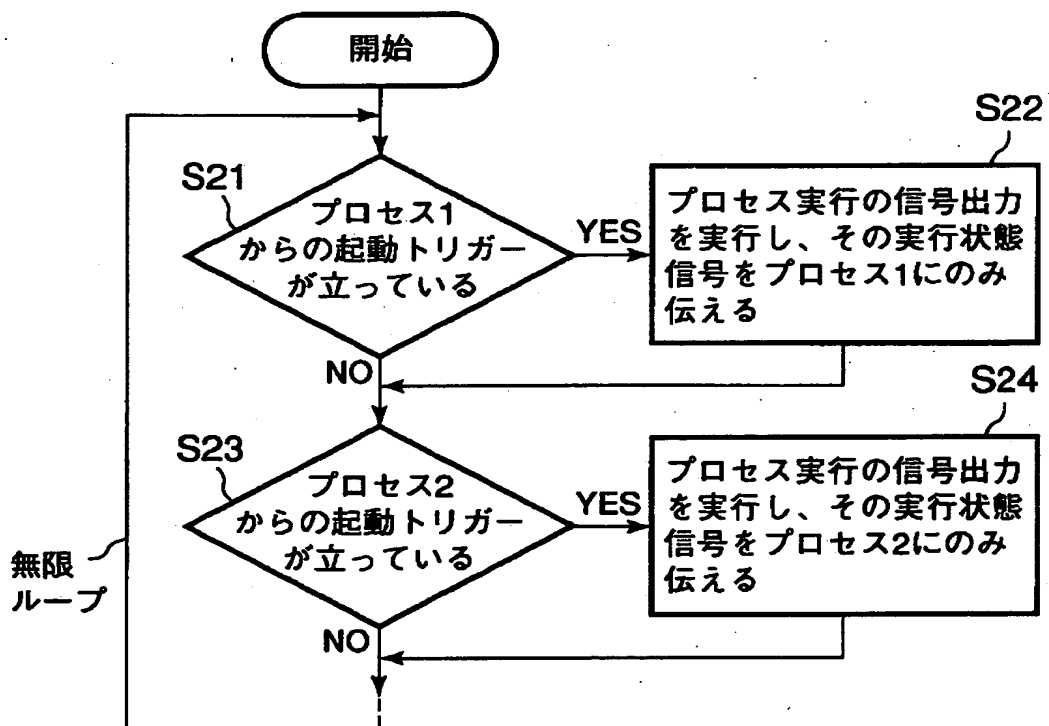
【図 1 3】



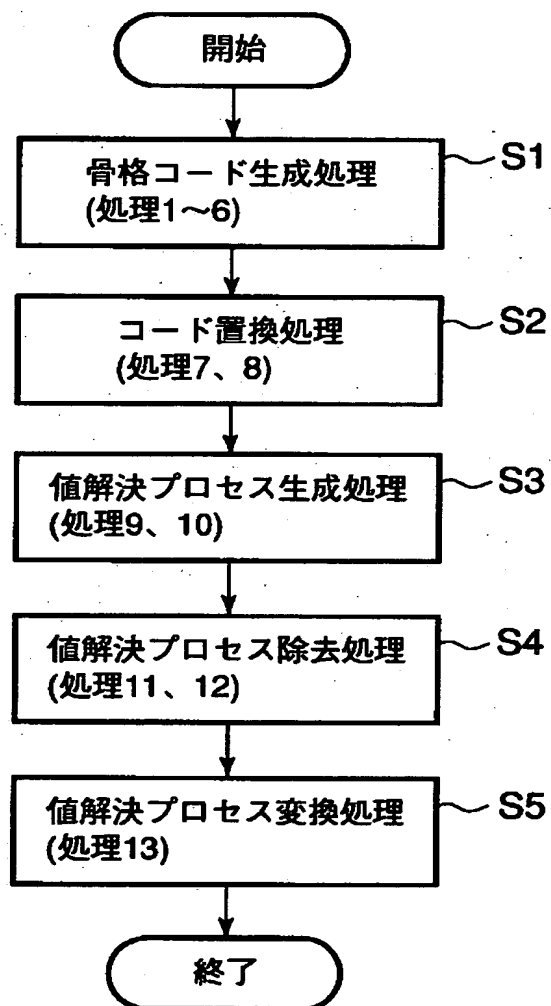
【図 14】



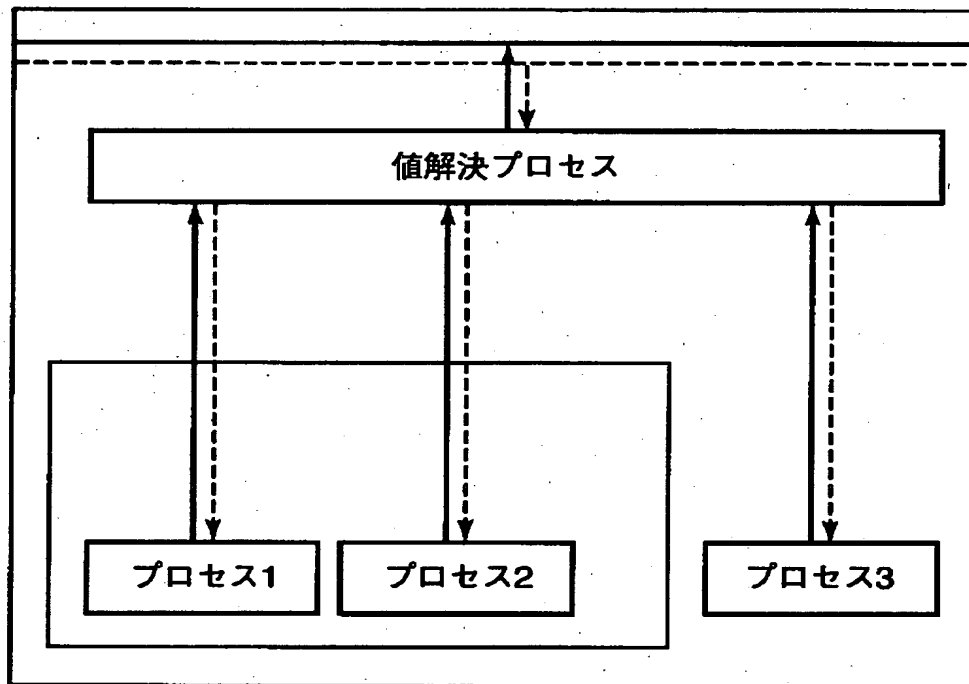
【図 15】



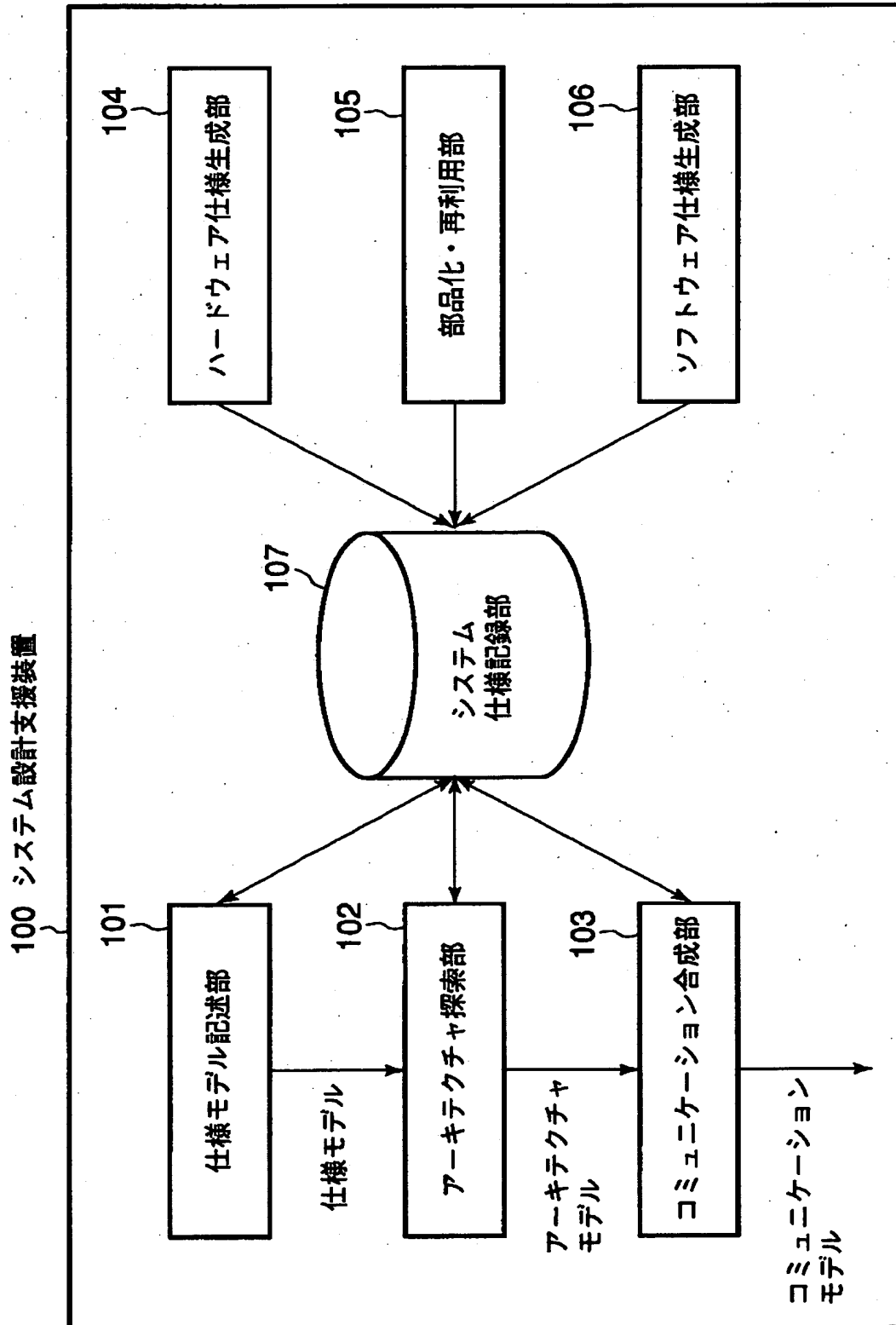
【図16】



【図 17】



【図18】



【書類名】 要約書

【要約】

【課題】 並列プログラムとその間に共有する変数を含むソフトウェア記述を同様の動作をする動作レベルのハードウェア記述言語記述に変換することのできるソフトウェア・ハードウェア言語モデル変換装置を提供すること。

【解決手段】 ソフトウェア記述言語による第1のモデルに同一の共有変数へ書き込みを行う複数の並列手続きが含まれているか否かを考慮せずに該第1のモデルをハードウェア記述言語による第2のモデルに変換する。これにより得られた第2のモデルに、同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する。検出された並列プロセスの全部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、共有変数の値を保持する信号へ出力する値解決プロセスを生成する。

【選択図】 図3

出 願 人 履 歴 情 報

識別番号 [000003078]

1. 変更年月日 1990年 8月22日
[変更理由] 新規登録
住 所 神奈川県川崎市幸区堀川町72番地
氏 名 株式会社東芝
2. 変更年月日 2001年 7月 2日
[変更理由] 住所変更
住 所 東京都港区芝浦一丁目1番1号
氏 名 株式会社東芝